



Modelling Moral Reasoning and Ethical Responsibility with Logic Programming

Fiona Berreby, Gauvain Bourgne, Jean-Gabriel Ganascia

► To cite this version:

Fiona Berreby, Gauvain Bourgne, Jean-Gabriel Ganascia. Modelling Moral Reasoning and Ethical Responsibility with Logic Programming. 20th International Conference, Logic for Programming, Artificial Intelligence, and Reasoning, Nov 2015, Suva, Fiji. pp.532-548, 10.1007/978-3-662-48899-7_37. hal-01395030v1

HAL Id: hal-01395030

<https://hal.sorbonne-universite.fr/hal-01395030v1>

Submitted on 25 Nov 2016 (v1), last revised 4 Oct 2017 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Modelling Moral Reasoning and Ethical Responsibility with Logic Programming

Fiona Berreby, Gauvain Bourgne, and Jean-Gabriel Ganascia

LIP6, Univ. Pierre and Marie Curie

4 Place Jussieu, 75005 Paris (France)

Email: {fiona.berreby, gauvain.bourgne, jean-gabriel.ganascia} @lip6.fr

Abstract. In this paper, we investigate the use of high-level action languages for representing and reasoning about ethical responsibility in goal specification domains. First, we present a simplified Event Calculus formulated as a logic program under the stable model semantics in order to represent situations within Answer Set Programming. Second, we introduce a model of causality that allows us to use an answer set solver to perform reasoning over the agent’s ethical responsibility. We then extend and test this framework against the Trolley Problem and the Doctrine of Double Effect. The overarching aim of the paper is to propose a general and adaptable formal language that may be employed over a variety of ethical scenarios in which the agent’s responsibility must be examined and their choices determined. Our fundamental ambition is to displace the burden of moral reasoning from the programmer to the program itself, moving away from current computational ethics that too easily embed moral reasoning within computational engines, thereby feeding atomic answers that fail to truly represent underlying dynamics.

1 Introduction

The study of morality from a computational point of view has attracted a growing interest from researchers in artificial intelligence; as reviewed in [?]. This endeavour can help us better understand morality, and reason more clearly about ethical concepts that are employed throughout philosophical, legal and even technological domains. Confronting ethical theories and philosophical works with the systematicity and logical constraints of programming languages indeed forces us to think about, and make explicit, the underlying mechanisms that characterize those works. It also sheds light on the possible inconsistencies or ambiguities that they may contain. In addition, as the autonomy of artificial agents grows and as an increasing amount of tasks are delegated to them, it becomes vital to equip them with the capacity to process moral restrictions and goals, be it within their own reasoning scheme or for interaction with human users.

The challenge therefore is to shift the burden of moral reasoning from the user or programmer to the program itself. Current works in computational ethics too often tend to embed moral factors within their computational engine, without

generating moral reasoning to speak of. The moral worth and causal implications of actions are atomically afforded, rather than extracted and ‘understood’ from facts and rules. In contrast, our aim is to provide a general and adaptable framework that enables the artificial agent to both understand the situation in which the dilemma arises and the ethical rules that constrain its actions, so as to determine from these only the correct course of action. To achieve this, we combine an entirely ethics-free model of the world with an ethical over-layer that the agent can understand and apply back onto its knowledge of the world. What is particularly important to note here is that at the centre of this process lies the notion of causality, for only when the agent can reason about causes and consequences can he begin to reason about moral choice and responsibility. Therefore, our model of moral reasoning and responsibility pivots around our discussion of causal models, which we implement in Event Calculus.

Formally, we chose the use of nonmonotonic logic as its study has been put forward by A.I. researchers as a way to handle the kind of defeasible generalisations that pervade much of our commonsense reasoning, and that are poorly captured by classical logic systems [?]. The term covers a family of formal frameworks devised to apprehend the kind of inference in which conclusions stay open to modification in the light of new information. On a regular basis it seems we draw conclusions from bodies of data that can be dropped when faced with new data. For example, we will hold that a certain bird can fly, until we learn that it is a penguin. This kind of default based reasoning is significantly present in ethical reasoning: we may determine the moral value of an action, for example theft, differently depending on surrounding information. Such factors as the presence of alternative options, indirect consequences, or extenuating circumstances might overthrow our ethical judgement. Accordingly, nonmonotonic goal specification languages are particularly well suited to modelling ethical reasoning.

The Doctrine of Double Effect (DDE) introduces fundamentally nonmonotonic precepts. It is a set of ethical criteria that was put forward most prominently by Philippa Foot for evaluating the ethical permissibility of an action that has both good and bad consequences [?]. It allows that while actions with negative consequences are morally prohibited a priori, there may be instances in which they are morally permissible. We therefore chose the DDE as the basis for moral recommendation, and applied it onto the dilemma of Trolley. Moreover, the DDE has been the focus of research in cognitive science, and was shown to be consistently corroborated by demographically different groups[?]; we aim to computationally explain but also test these intuitions. We begin by introducing the DDE and related current works in computational ethics, then lay out the basic tenets of ASP and the Event Calculus [Sects. 2 and 3]. Next, we expose our extension of the Event Calculus that enables us to handle issues pertaining to causal paths, and introduce our representation of the planning domain and problem [Sects. 4 and 5]. We then discuss the definitions and models of the notions of responsibility and prevention [Sect. 6], and describe the ethical motor and implementation of the Doctrine of Double Effect [Sect. 7], before concluding [Sect. 8].

2 Motivation

2.1 The Doctrine of Double Effect and The Trolley Problem

The DDE specifies four conditions that must be satisfied in order to render morally permissible an action that has both a good and a bad effect:

1. *Nature-of-the-act*. The action itself must either be morally good or indifferent.
2. *Means-end*. The good effect must not be reached by means of the bad effect.
3. *Right-intention*. Only the good effect must be intended, while the bad effect may only be an unintended side effect.
4. *Proportionality*. The good effect must be at least equivalent to the bad effect.

The DDE draws a distinction between intending harm and merely foreseeing harm, and can justify departures from purely consequentialist thinking in which only the proportionality condition operates. Significantly, it provides a reading of the Trolley Problem, an ethical dilemma formulated in Foot's 1967 paper. Consider the following scenario:

(*switch*) A train is running towards five workmen repairing train tracks. If the agent does nothing, the train will run over and kill them. However, the agent has the option of actioning a switch that will deviate the train off the tracks and onto side tracks along which one person is walking. This will kill that person.

Intuitively, respondents tend to agree that this action (actioning the switch), is ethically admissible [?]. This fits with the utilitarian notion that killing one person to save five is the better option (the other option being no action at all). Now take another case,

(*push*) There is no switch button, instead there is a bridge above the tracks on which stands an onlooker. Here, the agent knows that if they push the onlooker onto the tracks, the train will hit and kill the onlooker, stop as a result of the crash, and spare the five workmen.

Respondents have significantly deemed this action ethically impermissible [?], and are motivated by something other than utilitarian reasoning, since they choose the death of five over the death of one. The DDE successfully interprets this dilemma by justifying these seemingly inconsistent intuitions. Indeed, in the second case (*push*), while the nature-of-the-act and proportionality conditions are met, the means-end and right-intention conditions are violated: the death of the onlooker is used as a means to preventing the death of the five workmen, and as such is not just a foreseen side-effect but an intended act. About the nature-of-the-act condition it is important to note that only the intrinsic nature of the act itself is considered: even though *pushing someone off a bridge* is morally wrong, the act of *pushing* alone is not, unlike, for instance, stealing or lying. Looking back at the first case (*switch*), the death of the person walking on the other track plays no upstream causal role in the saving of the five: they are saved whether or not that one person dies, as long as the train leaves its original tracks.

2.2 Existing Works

In order to make clear the contribution of the present paper, it is necessary to first look at existing approaches in computational ethics before discussing our own. Pereira and Saptawijaya, in particular, also modelled the Trolley Problem and the DDE using prospective logic [?].

They represent the situation in which the agent throws the switch as follows:

```

turnSide ← consider(throwingSwitch).
kill(1) ← human(X), onSide(X), turnSide.
end(saveMen, niKill(N)) ← turnSide, kill(N).
observedEnd ← end(X, Y).

```

In parallel, the case in which the agent pushes a person on the tracks is modelled as follows:

```

onTrack(X) ← consider(shove(X)).
stopTrain(X) ← onTrack(X), heavy(X).
kill(1) ← human(X), onTrack(X).
kill(0) ← inanimateObject(X), onTrack(X).
end(saveMen, iKill(N)) ← human(X), stopTrain(X), kill(N).

```

In order to ascribe ethical criteria, they employ a priori constraints which rule out impermissible actions according to a particular ethical rule (corresponding to the means-end condition of the DDE) and a posteriori preferences that eliminate those solutions with worse consequences (the proportionality condition). The means end condition, importantly, is obtained via the two rules ‘*falsum* ← *intentionalKilling*.’ and ‘*intentionalKilling* ← *end(saveMen, iKill(Y))*.’

The difficulty with this kind of formalization is that it directly embeds the moral requirement into the model of the situation by indicating whether the killing is intentional (‘*iKill(N)*’), or not (‘*niKill(N)*’). The program is ‘told’ whether the outcome of the action fits with the ethical rules in place, through atomic statements of the form ‘*end(saveMen, iKill(N))* ← *human(X), stopTrain(X), kill(N)*’. This is problematic for a number of reasons. First, it fails to represent the actual reasoning that underpins moral decision making, in particular concerning what constitutes intentionality. Second, because it atomically specifies the ethical character of the situation’s outcome, it requires the creation of a different program for each new case. Therefore, even situations that share common features must be modelled independently, as is the case with trolley variations. This is redundant and can also lead to inconsistencies. Because rules lack expressive power, two identical expressions might refer to diverging stories, for example there is nothing in ‘*human(X), stopTrain(X), kill(N)*’ that indicates whether the killing is intentional or not, and as such could be employed in either case (here it is used for the push case). Moreover, there is no account of causality, such that the action and its consequences are not dynamically linked; the relationship between them is stated rather than inferred. Therefore, no account of ethical responsibility can be discussed on its basis. Finally, the model cannot logically confront ethical theories so as to make explicit their assumptions and give insight into them, nor can it enable us to explore and generate new ethical dilemmas for further testing. Even though it successfully points to

the expressive power of nonmonotonic logic for ethical reasoning, it seems that this account fails to "provide a general framework to model morality computationally" [?]. These remarks also apply to other current works in computational ethics, including models concerned with the Belief Desire Intention framework [?][?]. Instead, establishing an unchanging and ethics-free account of the world atop which can fit changeable ethical restrictions allows for generalisation, flexibility and automation. Separating the ethical constraints from the facts of the world is imperative if we are to model general ethical rules instead of performing case by case discrimination that resembles ethical judgement more than it does ethical theory.

3 Preliminaries

3.1 ASP

Answer Set Programming is a form of declarative logic programming suited for representing different Artificial Intelligence problems, particularly those that relate to knowledge representation and automated reasoning with incomplete information. Problems are encoded as *extended disjunctive programs*, finite logic theories from which are extracted stable models (answer sets) that declaratively identify the solutions to these problems [?].

We give here a very succinct overview of the answer set semantics for a program defined over a set of literals *Lit* (see [?] for more details). An *extended disjunctive program* (EDP) is a set of *rules* of the form: $L_1; \dots; L_l \leftarrow L_{l+1}, \dots, L_m, \text{not } L_{m+1}, \dots, \text{not } L_n$ ($n \geq m \geq l \geq 0$) where each $L_i \in \text{Lit}$ is a positive/negative literal, namely, A or $\neg A$ for an atom A . *not* is *negation as failure* (NAF), and *not* L is called an *NAF-literal*. The symbol ";" represents disjunction. For each rule r of the above form, $\text{head}(r)$, $\text{body}^+(r)$, $\text{body}^-(r)$, and $\text{not_body}^-(r)$ denote the sets of (NAF-)literals $\{L_1, \dots, L_l\}$, $\{L_{l+1}, \dots, L_m\}$, $\{L_{m+1}, \dots, L_n\}$, and $\{\text{not } L_{m+1}, \dots, \text{not } L_n\}$, respectively. A rule r is an *integrity constraint* if $\text{head}(r) = \emptyset$; and r is a *fact* if $\text{body}(r) = \emptyset$. A program P with variables is semantically identified with its ground instantiation. The semantics of EDPs is given by the *answer set semantics* [?]. A set $S \subseteq \text{Lit}$ *satisfies* a rule r if $\text{body}^+(r) \subseteq S$ and $\text{body}^-(r) \cap S = \emptyset$ imply $\text{head}(r) \cap S \neq \emptyset$. S satisfies a ground program P if S satisfies every rule in P . Let P be a program such that $\forall r \in P, \text{body}^-(r) = \emptyset$. Then, a set $S \subseteq \text{Lit}$ is a (consistent) *answer set* of P if S is a minimal set such that (i) S satisfies every rule from the ground instantiation of P , and (ii) S does not contain a pair of complementary literals L and $\neg L$. Next, let P be any EDP and $S \subseteq \text{Lit}$. For every rule r in the ground instantiation of P , the rule $r^S : \text{head}(r) \leftarrow \text{body}^+(r)$ is included in the *reduct* P^S if $\text{body}^-(r) \cap S = \emptyset$. Then, S is an *answer set* of P if S is an answer set of P^S .

3.2 The Event Calculus

The "Event Calculus" was first introduced in a 1986 paper by Bob Kowalski and Marek Sergot [?] as a logic programming framework used to represent and reason

about the effects of events or actions [?]. First employed in database applications, it has since then been integrated into other forms of logic programming, classical logic and modal logic, and used in wider contexts such as planning, abductive reasoning or cognitive robotics [?][?] [?].

The Event Calculus typically states that fluents (which are time-varying properties of the world) are true or false depending on whether they have been initiated or terminated by action occurrences. For the purpose of simplicity and to make the study of causality clearer, discrete time is employed, and is represented by integers. To fit the requirements of modeling ethical dilemmas pertaining to complex and realistic scenarios, one of our contributions has been to introduce automatic events in addition to actions. These automatic events occur when all their preconditions, in the form of fluents, hold, without direct input from the agent. Actions additionally require that the agent carries them out. As such, there are two types of events: actions and automatic events. We have made a further distinction between what we have called inertial fluents and non inertial fluents [?]. Once initiated by an event occurrence (or if initially true), inertial fluents remain true until they are terminated by another event occurrence. Non inertial fluents are only true at the point in time at which they have been initiated by an event occurrence, or at time T0 if they were true initially. An action is performed by the agent when the course of events begins, at time T0. A maximum of one automatic event can occur at each time point.

4 Adapted Event Calculus

Domain dependent axioms describe which events initiate and terminate which fluents, and which fluents are preconditions to which events. The dynamic domain given here is composed as follows: \mathcal{T} is a set of time points (as integers, variables T1, T2, T3...), \mathcal{P} is a set of positive fluents (variables P1, P2, P3...), \mathcal{F} is a set of fluents, composed of the items in the set P and of their negation (variables F1, F2, F3...), \mathcal{E} is a set of events (variables E1, E2, E3...). Our calculus is based on eight primary predicates. *Holds(F, T)* indicates that F is true at T; *Initially(F)* means that F is true at T0; *NonInertial(F)* points out the special kinds of fluents that are non inertial. *Occurs(E, T)* indicates that E occurs at T; *Automatic(E)* points out the special kinds of events that occur without direct agent input - all events that are not automatic are actions by default and require the agent's volition in addition to fluent preconditions. *Priority(E1, E2)* allows for prioritisation among automatic events but also among goals, i.e. between actions undertaken by the agent in cases where it is required to act more than once. *Effect(E, F)* expresses that F is an effect of E and *Precondition(F, E)* expresses that F is a precondition for E.

Fluents. In order to capture the behaviour of fluents relative to the occurrence of events, we define auxiliary predicates in terms of primary predicates. *Initiates(E, P, T)* indicates that E occurs and initiates P at T; *Terminates(E, P, T)* indicates that E occurs and terminates P at T; *Clipped(P, T)* indicates that a fluent which has been terminated by an occurrence of an event at T is clipped at T.

$$\begin{aligned}
& \textit{Initiates}(E, P, T) \leftarrow \textit{Effect}(E, P), \textit{Occurs}(E, T). \\
& \textit{Terminates}(E, P, T) \leftarrow \textit{Effect}(E, \textit{neg}(P)), \textit{Occurs}(E, T). \\
& \textit{Clipped}(P, T) \leftarrow \textit{Terminates}(E, P, T).
\end{aligned}$$

We can now axiomatize the principles that govern fluents. A fluent which has been initiated by an occurrence of an event at T, or was initially true, continues to hold until the occurrence of another event which terminates it. However, if it is non inertial, it holds at T only (or T0 if it was true initially). If it is not stated that a fluent holds, then its negation is true.

$$\begin{aligned}
& \textit{Holds}(P, T+1) \leftarrow \textit{Initiates}(E, P, T). \\
& \textit{Holds}(P, T0) \leftarrow \textit{Initially}(P). \\
& \textit{Holds}(P, T+1) \leftarrow \textit{Holds}(P, T), \textit{not Clipped}(P, T), \textit{not NonInertial}(P). \\
& \textit{Holds}(\textit{neg}(S), T) \leftarrow \textit{not Holds}(S, T).
\end{aligned}$$

Events. In order to capture the behaviour of events relative to the truth values of fluents (their preconditions), we first define a number of auxiliary predicates that constrain this mechanism. *MissingPrecondition*(F, E, T) means that there exists a precondition fluent F for E that does not hold at T; *Incomplete*(E, T) expresses that E is incomplete at T if it is missing one or more preconditions; *Possible*(E, T) expresses that E is possible at T if it is not missing any preconditions. *Overtaken*(E, T) expresses the fact that an event E has been overtaken by another event at T. This axiom must ensure that only one automatic event occurs at each time point, while an infinite number of events might be possible at any given time.

$$\begin{aligned}
& \textit{MissingPrecondition}(F, E, T) \leftarrow \textit{Precondition}(F, E), \textit{not Holds}(F, T). \\
& \textit{Incomplete}(E, T) \leftarrow \textit{MissingPrecondition}(F, E, T). \\
& \textit{Possible}(E, T) \leftarrow \textit{not Incomplete}(E, T). \\
& \textit{Overtaken}(E1, T) \leftarrow \textit{Possible}(E1, T), \textit{Possible}(E2, T), \textit{Priority}(E2, E1), E1 \neq E2.
\end{aligned}$$

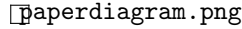
We can now axiomatize the principles that govern the occurrence of events. *Occurs*(E, T) denotes that E occurs at T if all its preconditions are true at T and no other event that has priority over it is also possible at T. We also specify that an event can only occur if it is possible.

$$\begin{aligned}
& \textit{Occurs}(E, T) \leftarrow \textit{Possible}(E, T), \textit{not Overtaken}(E, T), \textit{Automatic}(E). \\
& \leftarrow \textit{Occurs}(E, T), \textit{not Possible}(E, T).
\end{aligned}$$

5 Representing the Planning Domain and Problem

Defining the Domain and Problem. In order to represent the situation onto which ethical constraints are to be applied, we must hierarchize and specify a number of facts about the world we are aiming to represent. Within the present model, the simulation of a situation is characterised by **(a)** An initial situation that is composed by the truth values of all fluents at time T0; **(b)** A specification of the actions available to the agent, and of their causal powers over fluents; **(c)** A specification of automatic events, and of their causal powers over fluent; **(d)** A specification of the precondition fluents for events.

Representing the Trolley Problem. *Initial Situation.* Because our model aims at handling a general account of the world, it allows for the fact that both *switch* and *push* are actions that the agent can carry out in a single situational environment, rather than in two separate ones. As such, the initial situation states that all persons are alive, that there are 5 people on the 4th section of the main tracks and one person on the 3rd section of the side tracks (the sections of the tracks on which the people are stationed were chosen arbitrarily), and that the train is stationed on section 0 of the main tracks. There is a consequentialist specification in the DDE’s proportionality condition, namely that the good effect must be at least equivalent to the bad effect. Therefore, people who are involved in the dilemma are organised in numbered groups. Moreover, $on(train, B)$ is *NonInertial* since the train only stays on a section of the tracks for one time point.

 `paperdiagram.png`

$Initially(alive(G)).$	$NbPersons(group1, 5).$
$Initially(on(group1, main(4))).$	$NbPersons(group2, 1).$
$Initially(on(group2, side(3))).$	$NbPersons(group3, 1).$
$Initially(on(train, main(0))).$	$NonInertial(trainOn(B)).$
$Initially(on(group3, bridge)).$	

Event Preconditions and Effects. We determine the actions *push* and *switch*, and the automatic events *run* and *crash*. *Crash* has priority over *Run*; therefore the train stops as its preconditions for running are no longer fulfilled. In the following statements, \mathcal{N} refers to numbers, \mathcal{G} denotes groups of people, \mathcal{M} denotes sections of the main tracks, \mathcal{L} denotes sections of the side tracks, \mathcal{B} denotes both tracks ($\mathcal{B} = \mathcal{K} \cup \mathcal{L}$).

$Precondition(on(train, B), run(train, B)).$	$Precondition(on(G, bridge), push(G, B)).$
$Precondition(on(G, B), crash(G, B)).$	$Precondition(on(train, main(0)), switch)).$
$Precondition(on(train, B), crash(G, B)).$	
$Effect(push(G, B), on(G, B)).$	$Effect(crash(G, B), neg(on(train, B))).$
$Effect(switch, neg(on(train, main(0)))).$	$Effect(crash(G, B), neg(alive(G))).$
$Effect(switch, on(train, side(0))).$	$Priority(crash(G, B), run(B)).$
$Effect(run(train, main(N)), on(train, main(N+1))).$	
$Effect(run(train, side(N)), on(train, side(N+1))).$	

6 Modelling Responsibility

6.1 Agent Responsibility Regarding Caused Events

Defining Causation. Causality is a subtle notion that has been widely discussed in the philosophy literature, from Hume [?] to present day works [?][?][?]. But the challenge of defining causality reaches far beyond philosophy, and is for instance highly pertinent to legal decision making, such as in the event of a road accident in which legal responsibility must be determined. Causality is also central to the notion of ethical responsibility and decision making [?]. Agents are typically held responsible for (some of) their actions, but also for some external states of affairs that belong in the world. The question, when attributing

responsibility, is therefore to determine what these states of affairs are and why. As far as the causal powers of an agent's actions seem to constitute the only links between them and the world, it is natural to suggest that agents are responsible for those states of affairs *which they have caused* [?]. But there are also cases in which an agent can be held responsible for something *they didn't cause*, say by failing to rescue a drowning child. Here, the fact that they may be in a position to interact with the world and prevent a certain outcome still involves the notion of *causality*. As such, this notion must be investigated and defined.

Going back to Hume, we may be tempted to suggest a definition of causality in terms of counter-factual dependence: α is a cause of β , if, had α not happened, then β would not have happened. However, this naive definition fails to capture a number of subtleties present in causality, as it cannot deal with cases of pre-emption and over determination (when one cause may be replaced by another or when there are more causes than are necessary to produce the effect). Consider the following: *Suzy throws a rock at a bottle (s-throws), and shatters it (shatters). Billy was standing by with a second rock. Had Suzy not thrown her rock, Billy would have shattered the bottle by throwing his rock (b-throws)*[?]. Here, it is not the case that if (s-throws) had not happened, then (shatters) would not have happened, since (b-throws) would have made (shatters) happen. This definition therefore fails to capture the fact that something might have a cause while not being counter-factually dependent on it. This is particularly problematic when we want to address questions of responsibility: surely Suzy is responsible for shattering the bottle regardless of Billy's volitions.

Another branch of research has focused on structural causal models [?][?][?]. These have been particularly effective in assessing causal relationships between variables. However, while they handle well issues of counter-factual dependency, they fail to capture some of the intricacies emanating from the fact that causal relations hold in dynamically changing situations [?]. In particular, they cannot distinguish between conditions and transitions, or between actions and omissions [?], yet these distinctions are central to the study of responsibility (for instance, causal weight is not equally divided between acting and omitting to act). Reversely, as argued by Hopkins and Pearl in [?], the formal semantics of Situation Calculus succeed in handling these issues. This is also true of Event Calculus, and further motivates our choice of this particular formalism. The fluent/event distinction in Event Calculus allows us to model the condition/transition divide, and the facts that *events occur* and *fluents hold or fail to hold* provide the adequate tools for addressing the action/omission distinction. These formal objects that correspond in natural ways to the situations at hand. For more, see [?]. In the next section, we present our corresponding account of causation.

Modelling Causation. We consider a fluent P to be a consequence of an event $E1$ if $E1$ initiates P (or terminates $\text{neg}(P)$), regardless of whether another event $E2$ would have initiated P in the absence of $E1$. Likewise, an event $E1$ is a consequence of a fluent P if P is a precondition to $E1$, and both are true. This accommodates for the possibility that there may be more than one precondition

for the occurrence of E1, and that P be not considered a cause of E1 if E1 does not occur (say because the other preconditions were not fulfilled). Our model considers the causal links that hold between events and fluents (i.e. the consequences that these would have on the world were they to obtain), separately from whether they actually obtain. This, and our choice to define causality in terms of consequences, affords us with a useable trace of causal paths and allows us to dynamically assess causal relationships.

We first define a domain \mathcal{D} such that $\mathcal{D} \equiv \mathcal{F} \cup \mathcal{E}$. *Consequence*(D1, D2) indicates that D2 is a consequence of D1. The reflexivity of consequences is necessary to simplify the definitions of predicates that contain the *Consequence* predicate. For the sake of simplifying rules that pertain to causal chains, all fluents and events that hold or occur are also considered consequences of themselves. As such:

$$\begin{aligned} \text{Consequence}(E, F) &\leftarrow \text{Effect}(E, F), \text{Occurs}(E, T), \text{Holds}(F, T+1). \\ \text{Consequence}(F, E) &\leftarrow \text{Precondition}(F, E), \text{Holds}(F, T), \text{Occurs}(E, T). \\ \text{Consequence}(D1, D3) &\leftarrow \text{Consequence}(D1, D2), \text{Consequence}(D2, D3). \\ \text{Consequence}(D, D) &\leftarrow \text{Holds}(D, T). \\ \text{Consequence}(D, D) &\leftarrow \text{Occurs}(D, T). \end{aligned}$$

Now that we have established a model of causality, we can see that formulating the bottle example in terms of an Event Calculus model, rather than via counterfactual dependence, is unproblematic: If Suzy throws and hits the bottle, then she is considered responsible for its shattering. If she doesn't throw, and it is specified that 'b-throw \leftarrow not s-throw', then Billy will throw and be considered responsible.

6.2 Agent Responsibility Regarding Prevented Events

Defining Prevention. Ethical responsibility is most often associated with the occurrence of events, for example pertaining to the number of deaths caused by an air strike or the amount of aid given to a relief centre. Yet agent responsibility is equally a question of avoided or prevented harms; think of lives saved by a particular military strategy or medical investment, or of people rescued from falling beneath the poverty threshold. Works in the computational ethics literature often fail to even address this fact, or consider prevention uniformly with causation [?][?]. However, these two concepts rely on severely different mechanisms, and make different computational demands. In particular, unlike causality which is concerned with the actual state of affairs of the world, prevention requires that we be able to represent possible, but untrue worlds: we must account for what *could* have happened - but didn't. We must be able to say *why* it didn't happen, and whether the agent is truly responsible for this. Thus, to model the fact that an agent prevents an event from occurring by performing an action A, we must be in a position to compare the actual chain of events with the hypothetical chain of events in which the agent does not perform A. Within ASP, one way to achieve this is to simulate both cases and compare the results, however, this solution requires post processing the individual answer sets. The action theory architecture that specifies preconditions for events and fluents allows us to avoid

this procedure, and provides us with the traceable account of causal paths needed to model ethical responsibility.

An event E1 prevents an event E2 if all three of the following are true: **(a)** E1 terminates a fluent F that is a precondition to an event E3 of which E2 is a consequence (note that it is possible that $E2 \equiv E3$); **(b)** all other preconditions of E2 hold; **(c)** E2 does not occur. (a) ensures that the E1 may break the causal chain between E1 and E2 at any point. For example E1 may impede a precondition to E2, or impede the precondition to a precondition to E2. (b) ensures that E2 would have happened had E1 not happened: it guarantees the counter-factual dependency of E2 on $\neg E1$. (c) ensures that if E2 occurs as a result of being caused by another event through another causal path, then E1 cannot be said to succeed in preventing E2.

Modelling Prevention. We model the *Prevents* predicate, which accounts for the causal relations that exist but that have not been executed. We define the predicate *HypConsequence* ($D1, D2$), which denotes that a fluent and an event are causally linked, but says nothing about the actual state of the world, i.e., about whether this causal link has been instantiated. Hypothetical consequences, like consequences, are transitive (and reflexive).

$$\begin{aligned} HypConsequence(E, F) &\leftarrow Effect(E, F). \\ HypConsequence(F, E) &\leftarrow Precondition(F, E). \\ HypConsequence(D1, D3) &\leftarrow HypConsequence(D1, D2), HypConsequence(D2, D3). \\ HypConsequence(D, D) &. \end{aligned}$$

Next, we define a number of prior predicates: *TransTerminates*($E1, F2$) denotes that an event E transterminates a fluent F2 if it terminates a fluent F1 that is causally linked, and causally anterior to F2. This definition allows for indirect cases where E affects a non contiguous fluent further down the causal chain. *NotPrevents*($E1, E2$) identifies the cases in which an event E1 causes the termination of a precondition fluent to an event E2, but where at least one *other* precondition for E2 is missing (i.e., one that has not been transterminated by E1). Finally, in order to preclude the possibility that the event occurs via another causal path, it is necessary to define the *Happens*(E) predicate that characterises any event that has occurred at some point in time.

$$\begin{aligned} TransTerminates(E, F2) &\leftarrow Terminates(E, F1, T), HypConsequence(F1, F2). \\ NotPrevents(E1, E2) &\leftarrow TransTerminates(E1, F1), Precondition(F1, E2), MissingPrecondition(F2, E2, T1), not TransTerminates(E1, F2), F1 \neq F2. \\ Happens(E) &\leftarrow Occurs(E, T). \end{aligned}$$

We can now define the pivot predicate *Prevents*($E1, E2$), which states that an event E1 prevents and event E2 if E1 transterminates a precondition for E2, all other preconditions for E2 hold and E2 does not happen.

$$Prevents(E1, E2) \leftarrow Occurs(E1, T), TransTerminates(E1, F), Precondition(F, E2), not NotPrevents(E1, E2), not Happens(E2).$$

7 Ethical Implementation

7.1 Determining the Desirability of Events

In order to implement the DDE, which places additional ethical valuation on actions with already determined desirable and undesirable effects, we must first have an account of which events are desirable and which are not. One way of doing this is by simply stating that, for instance, a train crash is undesirable and that people staying alive is desirable. Within our semantics, however, it is fitting to evaluate the desirability of events in terms of the effect they have on ethically relevant fluents. Indeed, an event can only be measured relative to the effect it has on the world. For instance, a collision is significant only in so far as it changes the state or condition of the parties involved in it. We therefore are interested in the moment at which events, atomically and independently of surrounding factors, become desirable or undesirable.

Rights-based ethical theories are particularly well suited to this task, as they make moral claims over the permissibility of actions depending on whether these respect certain rights, which can be likened to states of affairs, such as, for example, the right *to property* or the right *to safety* [?][?]. We base ourselves on Beauchamp and Childress’s account of a right which they define as a “*justified claim that individuals and groups can make upon other individuals or upon society; to have a right is to be in a position to determine by one’s choices, what others should do or need not do*” [?]. This definition captures well the fact that a right denotes both a state of affairs for the person concerned (the exercise of the right) and a constraint on others (which they can respect or violate through their actions). We therefore chose to define an undesirable event as one that clips (violates) a right, which is here a special kind of fluent. We specify the corresponding situational domain \mathcal{R} as a set of rights, and the definitional axioms:

$$\begin{aligned} \text{Undesirable}(E) &\leftarrow \text{Effect}(E, \text{neg}(R)). \\ \text{Desirable}(E) &\leftarrow \text{Effect}(E, R). \\ \text{Indifferent}(E) &\leftarrow \text{not Desirable}(E), \text{not Undesirable}(E). \end{aligned}$$

7.2 The Doctrine of Double Effect.

The Nature-of-the-act Condition. The first axiom of the DDE is modelled as such: $\text{Impermissible}(A) \leftarrow \text{not Desirable}(A), \text{not Indifferent}(A)$.

The Means-end and the Right-intention Conditions. These two axioms are collapsed into one rule within our model, for we consider that using an event as a means to an end (i.e. for the occurrence or prevention of another event), is equivalent to intending that event. Correspondingly, unintended side effects cannot be means to ends. It follows that the two axioms are analogous computationally, unless intentions are explicitly modelled. A good effect may be reached in one of two ways, either by causing a desirable event or by preventing an undesirable one. The reverse is true for bad effects, therefore, four rules must be specified.

$$\begin{aligned}
& \text{Impermissible}(A, T) \leftarrow \text{Occurs}(A, T), \text{Consequence}(A, E1), \text{Consequence}(E1, \\
& E2), \text{Undesirable}(E1), \text{Desirable}(E2). \\
& \text{Impermissible}(A, T) \leftarrow \text{Occurs}(A, T), \text{Consequence}(A, E1), \text{Prevents}(E1, E2), \\
& \text{Undesirable}(E1), \text{Undesirable}(E2). \\
& \text{Impermissible}(A, T) \leftarrow \text{Occurs}(A, T), \text{Prevents}(A, E1), \text{Prevents}(E1, E2), \\
& \text{Desirable}(E1), \text{Undesirable}(E2). \\
& \text{Impermissible}(A, T) \leftarrow \text{Occurs}(A, T), \text{Prevents}(A, E1), \text{Consequence}(E1, E2), \\
& \text{Desirable}(E1), \text{Desirable}(E2).
\end{aligned}$$

The Proportionality Condition. The fourth axiom of the DDE introduces a consequentialist requirement, as it demands the weighing against each other of the action’s good and bad effects. There are numerous ways in which effects can be measured, both quantitatively and qualitatively. We chose to gauge events in terms of the number of people involved in them and define the predicate $\text{Weight} \subseteq \mathcal{E} \times \mathcal{N}$, determined for each event by rules of the type “ $\text{Weight}(\text{crash}(G, B), N) \leftarrow \text{NbPersons}(G, N)$ ”. As such, performing an action A at T is not permissible if it causes two automatic events E1 and E2, which respectively involve N1 and N2 numbers of people, and where E1 is undesirable and E2 is desirable, if N1 is greater than N2.

$$\begin{aligned}
& \text{Impermissible}(A, T) \leftarrow \text{Occurs}(A, T), \text{Consequence}(A, E1), \text{Consequence}(A, \\
& E2), \text{Undesirable}(E1), \text{Desirable}(E2), \text{Weight}(E1, N1), \text{Weight}(E2, N2), N1 > N2.
\end{aligned}$$

Because of the combination of causation and prevention, as with the means-end condition, three other axioms are also necessary to represent every possible situation that might result from an action with both a good and bad effect, corresponding to the Consequence x Prevents matrix (as with the previous condition). In the model, if we reverse the number of people on the side and main tracks, the switch action becomes impermissible, since more people will die if the agent pushes the switch. The modular nature of the model allows us to play around with the characteristics of the dilemma and explore the DDE.

7.3 Ethical Choice

The agent selects one and only one action (here either *push* or *switch*) to carry out in separate scenarios, resulting in different answer sets, then only chooses to perform the actions that are not impermissible.

$$\begin{aligned}
& 1\{\text{Occurs}(\text{push}(\text{group3}, \text{main}(2)), T), \text{Occurs}(\text{switch}, T)\} 1 \leftarrow T=0. \\
& \leftarrow \text{Occurs}(A, T), \text{Impermissible}(A, T).
\end{aligned}$$

We are then left with no, one or a number of stable models that each represent an action that is permissible in regards to the specified situation and the ethical rules that regulate the agent’s behaviour. In our model of the Trolley Problem and consistently with experimental findings [?], the unique stable model represents the choice of the *switch* action (see Appendix A). Note that for now, this is a mostly non disjunctive stratified problem that could be programmed with logic programming tools other than ASP. However, the extensions we envisage for the model will make full use of the properties of ASP, in particular regarding ethical plans.

8 Conclusion

Computing ethical theories allows us to reach a greater understanding of the concepts at play, both formally in relation to the predicates and formalisms employed, and in relation to notions used by philosophers and law-makers. Using ASP to model the DDE has shed light on the importance and difficulty of handling causal paths in order to justify claims of ethical responsibility. In particular, it has exposed the necessity to tackle both caused and prevented events. Moreover, in the case of prevention, it has made clear the requirement of adequately handling situational circumstances: an agent can only prevent an event that would have occurred had he not acted. This underlines the fact that responsibility concerns not just the effect of actions, but is to be apprehended from the state of the world itself. While these remarks belong in the realm of common sense for human agents, they are remarkably heavy in repercussion for the modelling of autonomous agents faced with ethical challenges.

The model we have presented here adapts the Event Calculus to facilitate the examination and resolving of ethical dilemmas within a nonmonotonic logic domain. While its present focus is on the Trolley Problem and the DDE, its scope is extensive and adaptable. In order to develop our model, we therefore envision a number of future avenues. First, we believe that we need to further explore ways of expressing intentionality so as to enable artificial agents to evaluate their own moral choices as well as those of others. This will enable the study of other ethical theories that are concerned with agent intention, such as the Doctrine of Triple Effect, put forward by Kamm as a response to dilemmas that the DDE fails to properly handle [?][?]. Staying within the Trolley Problem, we are also currently working on the generation of ethical dilemmas based on the situational domain. The aim of this is to test the DDE on numerous and novel variations of the problem, with the possible adjunction of new empirical data. Creating new dilemmas opens avenues for testing ethical theories and refining them. Finally, we intend to model different ethical criteria such as Kant's categorical imperative or value-based ethics, thereby extending our framework to other ethical traditions and applicative domains. This might also include the formulation of ethical plans of action, working up towards a true planning domain.

A Appendix A

The rules '1{occurs(push(group3, main(2)), T), occurs(switch, T)}1 :- T=0.' and ':- occurs(A, T), impermissible(A, T).' generate one stable model that corresponds to the *permissible* switch action.

```
initiates(switch, on(train, side(0)), 0).
initiates(run(train, main(0)), on(train, main(1)), 0).
initiates(run(train, side(0)), on(train, side(1)), 1).
initiates(run(train, side(1)), on(train, side(2)), 2).
initiates(run(train, side(2)), on(train, side(3)), 3).
```

```

occurs(switch, 0).
occurs(run(train, main(0)), 0).
occurs(run(train, side(0)), 1).
occurs(run(train, side(1)), 2).
occurs(run(train, side(2)), 3).
occurs(crash(group2, side(3)), 4).
overtaken(run(train, main(1)), 1).
overtaken(run(train, side(3)), 4).
prevents(switch, crash(group1, main(4))).
terminates(switch, on(train, main(0)), 0).
terminates(crash(group2, side(3)), alive(group2), 4).
terminates(crash(group2, side(3)), on(train, side(3)), 4).
permissible(switch, 0).

```

Disabling the second rule (`:- occurs(A, T), impermissible(A, T).`) allows us to look at the stable model for the *impermissible* push action:

```

initiates(push(group3, main(2)), on(group3, main(2)), 0).
initiates(run(train, main(0)), on(train, main(1)), 0).
initiates(run(train, main(1)), on(train, main(2)), 1).
occurs(run(train, main(0)), 0).
occurs(push(group3, main(2)), 0).
occurs(run(train, main(1)), 1).
occurs(crash(group3, main(2)), 2).
overtaken(run(train, main(2)), 2).
prevents(crash(group3, main(2)), crash(group1, main(4))).
terminates(crash(group3, main(2)), alive(group3), 2).
terminates(crash(group3, main(2)), on(train, main(2)), 2).
impermissible(push(group3, main(2)), 0).

```